Newton's Interpolation Algorithm Implemented as an Upgraded Petri Net Model

Perica S. Štrbac, Slobodan Obradović
School of Electrical and Computer Science
Belgrade, Serbia
strbac68@gmail.com
slobodan.obradovic@viser.edu.rs

Nikola Davidović
Faculty of electrical engineering
East Saraevo, RS, BH
nikola.davidovic@etf.unssa.rs.ba

Abstract — **The objective of this paper is modeling, simulation and analysis of Upgraded Petri Net (UPN) model which implements Newton's interpolation algorithm to show parallelism and synchronization at register transfer level. The paper includes UPN theory and the UPN model of Newton's interpolation algorithm for given input data as a set of data points. The model refers to: setting input data (data points and value of variable x), parallel execution of the Upgraded Petri net model with synchronization between these parallel executions and generating output data (y=f(x)). The paper also represents simulation and analysis of the UPN model. The executions of UPN are based on parallel firing of a group of transitions. Original software for modeling and simulations of UPN, PeM (Petri Net Manager), is used for all models described in this paper. The suitability of UPN for modeling the Newton's interpolation algorithm for given input points is examined and established.**

Key words — Upgraded Petri Net; Newton's Interpolation; Modeling; Simulation; Analysis; Algorithm;

I. INTRODUCTION

This paper presents the usage of Upgraded Petri nets in the example of modeling, simulation and analysis of a genetic algorithm which solving Knapsack problem.

Upgraded Petri nets are a formal mathematical apparatus which enables modeling, simulation and process analysis [1] through interactive monitoring of Petri net execution from the initial phase, all the way to the final version. The UPN was developed for simulation and analysis of processes, particularly at the register transfer level. The hierarchical structure of an UPN enables that a model consists at the same time of elaborate pieces essential for the analysis at a certain level, and also of some general pieces whose details are irrelevant for the analysis at the given level of abstraction [2]. The UPN is an extension of an ordinary Petri Net and a formal modeling tool appropriate for simulation and analysis of processes, particularly at the register transfer level (RTL) [3].

Newton's interpolation algorithm are based on calculating polynomial coefficients by using divided differences. Main goal is to find appropriate polynomial (i.e. its coefficients) which passes through given input points. By using this polynomial we can calculate its value for data which belongs to the interval determined by given input points. [4].

In this paper we show and analysis an UPN model of Newton's interpolation algorithm with respect to parallelism and synchronization which is implemented into the model [4]. This parallelism is realized through parallel execution of the UPN net based on parallel firing of set of enabled transitions. The model can generate value of y coordinate for given input points and given value of x coordinate of the point we want to interpolate by using this polynomial.

In the UPN model we use input function, output function, marking function, transition function, place attribute function, transition firing level and rules for UPN net execution. Initial marking refers to given input points which includes their x and y coordinates and value of the x coordinate of the point that we want to interpolate. The |UPN model through its execution calculates appropriate coefficient of the interpolation polynomial then calculates value of the y coordinate of the interpolated point.

II. AN UPN FORMAL THEORY

Upgraded Petri nets formal theory is based on functions [3]. Upgraded Petri net is a 9-tuple:

$C = (P, T, F, B, \mu, \theta, TF, TFL, PAF)$

where:

$P=\{p_1, p_2, p_3, ..., p_n\}, n>0$
- a finite nonempty set of places p_i

$T=\{t_1, t_2, t_3,...,t_m\}, m>0$
- a finite nonempty set of transitions t_j

$F: T\times P\rightarrow N_0$ - Input Function;
$B: T\times P\rightarrow N_0$ - Output Function;
$\mu: P\rightarrow N_0$ - Marking Function:
$\theta: T\times\Delta\rightarrow\lambda$ - Timing Function;
$TF: T\rightarrow A$ - Transition Function;
$TFL: T\rightarrow N_0$ - Transition Firing Level;
$PAF: P\rightarrow(x, y)$ - Place Attributes Function;

The input function assigns a non-negative number to an ordered pair $(t_i,p_j)\in T\times P$. The assigned non-negative number defines how many times the place pi is input as compared to the transition t_i. N_0 represents the set of non-negative integers. The set of places which are input as compared to the transition t_j is presented as follows $*t_j=\{ p_i\in P, F(t_j, p_i)>0\}$. For the

presentation of the place $p_i \in {}^*t_j$ which have the standard input compared to the t_j, the sign ${}^*t_j^S$ will be used, and sign $F^S(t_j, p_i)$ will be used for such input function. For the places $p_i \in {}^*t_j$ with inhibitor input in relation to the t_j transition, the sign ${}^*t_j^I$ will be used and sign $F^I(t_j, p_i)$ for the input function.

The output function gives a non-negative integer to the ordered pair (t_i,p_j). The assigned non-negative integer shows how many times the place p_i is input in relation to the t_i transition. The set of places which are input in relation to the t_j transition is presented as follows $t_j^* = \{ p_i \in P, B(t_j, p_i) > 0 \}$. The marking function assigns a non-negative integer to the pi place.

The marking function can be defined as n-dimension vector (marking): $\mu=(\mu_1,\mu_2,...,\mu_n)$, where $n=|P|$. Instead of sign μ_i it can be used the sign $\mu(p_i)$.

The timing function θ assigns the probability $\lambda_{ij} \in [0,1]$ to an ordered pair $(t_i, j) \in T \times N_0$, i.e., $\lambda_{ij} = \theta(t_i, j)$. The transition function gives an operation $\alpha_j \in A$ to the t_j transition. Sign A is the set of operations which can be assigned to the transition.

The transition firing level of the transition gives a non-negative integer to the transition t_j. If this number not equals zero, it shows the number of $p_i \in {}^*t_j$ places takes part in the transition firing, and if this number equals zero, then all the places $p_i \in {}^*t_j$ affect the t_j transition firing.

Place attributes function assigns an ordered pair (x,y) to the place p_i. The x component is a real number called x attribute, and y is a non-negative integer called y attribute (i.e. $x \in R$, $y \in N_0$). Over the x attribute belonging to the $p_i \in {}^*t_j$ places, the α_j operation assigned to the t_j transition executes where the order of operands in the operation α_j is defined by the y attributes which belong to the p_i place in accordance to TFL function take part in the transition firing.

An Operation Assigned to a Transition: Function TF assigns to a transition t_j one operation. This operation can be: arithmetical operation, logical operation or file operation [1].

Inside the suite *PeM* a file which is a target of file operation function has an **.mem* extension. This **.mem* file is a text file and it is used for simulation of computer system memory. One line inside the **.mem* file refers to context of one memory location of computer system that we are modeling.

An arithmetical operation $\alpha_j \in A$, which is assigned to the transition $t_j \in T$, uses attributes x which belong to the places $p_i \in {}^*t_j$ as operands of that operation. A result of an arithmetical operation $\alpha_j \in A$ will be placed into the attributes x which belong to the places $p_i \in t_j^*$. The order of an operand (i.e. order of attributes x which belong to the places $p_i \in {}^*t_j$) in an arithmetical operation $\alpha_j \in A$ is defined by attributes y which belong to the places $p_i \in {}^*t_j$. A logical operation $\alpha_j \in A$ which is assigned to the transition $t_j \in T$, uses attributes x which belong to the places $p_i \in {}^*t_j$ as operands of that operation. If a result of the logical operation $\alpha_j \in A$ is logical false the transition $t_j \in T$ is disabled and will stay in that state until the result of this logical operation $\alpha_j \in A$ becomes logical true. The order of an operand (i.e. order of attributes x which belong to the places $p_i \in {}^*t_j$) in a logical operation $\alpha_j \in A$ is defined by attributes y which belong to the places $p_i \in {}^*t_j$. A file operation $\alpha_j \in A$ which is assigned to the transition $t_j \in T$ performs over the context of a file which extension is equal to **.mem*. A File Operation $\alpha_j \in A$ addresses context of a **.mem* by using attribute x which belongs to the places $p_i \in {}^*t_j$. A result of this operation $\alpha_j \in A$ changes the value of attributes x which belong to the places $p_i \in t_j^*$. The result also can change attributes y which belong to the places $p_i \in t_j^*$, or can change context of addressed line into the **.mem* file.

An UPN Graph: Upgraded Petri Net is represented via formal mathematical apparatus or graphically. An UPN is represented by bipartite multigraph as is in Petri-net.

An Upgraded Petri Net executing represents change of system state from the current state to the next state. This migration from one state to the other one is triggered by firing of the transitions. By UPN executing: marking vector can be changed, contents of **.mem* file can be changed, and attributes which belong to the places $p_i \in t_j^*$ of enabled transition t_j can be changed.

A transition $t_j \in T$ can be enabled in Upgraded Petri Net: $C = (P, T, F, B, \mu, \theta, TF, TFL, PAF)$ if the next 3 conditions are satisfied:

1° If the timing function $\lambda_{jk} = \theta(t_j, k) > 0$; (1)

2° If $TFL(t_j)>0$ then $(\#p_i(S)) + (\#p_i(I)) = TFL(t_j)$, (2)
and if $TFL(t_j)=0$ then $(\#p_i(S)) + (\#p_i(I)) = |{}^*t_j|$,
where
$\#p_i(S)$ is a number of places $p_i \in {}^*t^S_j$ such that
$\mu(p_i) \geq F^S(t_j, p_i)$, and
$\#p_i(I)$ represents a number of places $p_i \in {}^*t^I_j$ for which
$\mu(p_i) = 0$;

3° If a logical operation $\alpha_j \in A$ assigned to the (3)
transition t_j, then the result of the operation
α_j must be equal to true.

A marking vector μ will be changed to new marking vector μ' by firing of transitions t_j, where:

$\mu'(p_i) = \mu(p_i) - F(t_j, p_i) + B(t_j,p_i)$, for places $p_i \in {}^*t_j^S$
$\mu'(p_k) = \mu(p_k) + B(t_j,p_i)$, for places $p_k \in {}^*t_j^I$

By firing of the transition t_j an arithmetic operation is executed or a file operation is executed with respect to the operation that is assigned to t_j by function $TF(t_j)$. A logical operation which assigned to the transition tj by function $TF(t_j)$ will be executed if conditions (1) and (2) related to t_j are equal to true.

A conflict in Upgraded Petri Net influences UPN executing. A conflict in UPN is the same as the conflict in Petri-net.

An UPN reachability tree graphically represents all possible marking vectors which can occurs during an UPN execution for given initial marking. Reachability tree shows all states which model can reach from the initial state. The UPN reachability tree is the same as the Petri Net reachability tree.

An UPN executing refers to a concurrent firing of the enabled. An UPN execution generates an UPN flammability tree. This tree is such tree where a node of the tree is a set of the transitions which are enabled at the same time. If there is the same node as the current node in the flammability tree then generating of the flammability tree will be stopped. There are four types of nodes in a flammability tree: root node, double node, dead node, and inner node.

III. THE UPN MODEL: PARALLELISM AND SYNCHRONIZATION AT RTL LEVEL OF NEWTON'S INTERPOLATION ALGORITHM

In this section an UPN model which refers to Newton's interpolation algorithm is described [5], [6], [7]. This UPN model is based on parallelism and synchronization [8], [9].

For given set of different k+1 (input) points Newton's interpolation polynomial is:

$$N(x) = \sum_{j=0}^{k} a_j n_j(x) \qquad (4)$$

where:

$$n_j(x) = \prod_{i=0}^{j-1} (x - x_i) \qquad (5)$$

$$a_j = \left[y_0, ..., y_j \right] \qquad (6)$$

are Newton basis polynomial and coefficients (divided differences), respectively.

The initial marking of the UPN model which refers to Newton's interpolation algorithm is shown in Figure 1. Initial marking is $\mu_{(p\text{-}1)}=3$, $\mu_{(p\text{-}2)}=4$, $\mu_{(p\text{-}3)}=4$, $\mu_{(p\text{-}4)}=3$, $\mu_{(p\text{-}4)}=2$, $\mu_{(p\text{-}5)}=2$, $\mu_{(p\text{-}6)}=2$, $\mu_{(p\text{-}7)}=1$, $\mu_{(p\text{-}32)}=3$. An attribute *X* of the places *p-1, p-2, p-3, p-4* refer to *x* coordinates of the given points. An attribute *X* of the places *p-5, p-6, p-7, p-8* refer to y coordinates of the given points. At last, attribute *X* of the place *p-32* refers to value of the *x* coordinate of interpolated point.

At this moment the transitions *t-1, t-2, t-5, t-4, t-8, t-7, t-19, t-20, t-17, t-29, t-25* and *t-34* are enabled and can be fired simultaneously. By firing of the transitions *t-1, t-2, t-5, t-4, t-8* and *t-7* the attributes *X* of the places *p-9, p-10, p-12, p-11, p-14, p13* refer to *y1-y2, x1-x2, y3-y2, x3-x2, y3-y4, x3-x4,* respectively. By firing of the transition *t-19, t-20, t-29* and *t-34* value *x1:X, x2:X, x3:X* and *x3:X* will be copied into *p-27:X, p-28:X*, *p-36:X* and *p-44:X*, respectively. These copies has different *Y* attributes with respect to original (*original:Y=1 copy:Y=2, original:Y=2 copy:Y=1*) except for place *p44*. This is necessary in this model because of appropriate order of operand of transition function, such as subtraction (*SUB*). By firing of the transition *t-17* and *t-25* the attributes *X* of the places *p-24* and *p-34* refer to *x4-x1, x-x2,* respectively.

After parallel firing of the transitions *t-1, t-2, t-5, t-4, t-8, t-7, t-19, t-20, t-17, t-29* and *t-25* the new set of transitions can be fired simultaneously. This set includes the transitions as follows: *t-3, t-6, t-9, t-10, t-13, t-24* and *t-30*. By parallel firing of the transitions *t-3, t-6* and *t-9,* the attributes *X* of the places *p-17 (A), p-16 (B)* and *p-15 (C)* refer to (*y1-y2)/(x1-x2), (y3-y2)/(x3-x2), (y3-y4)/(x3-x4),* respectively. After parallel firing of the transitions *t-10, t-13, t-24* and *t-30* the attributes *X* of the places *p-19, p-22, p-31* and *p-38* refer to *(x3-x1), (x2-x4), (x-x1)* and *(x-x3),* respectively.

At this moment the set of the transitions: *t-11, t-22, t-23* and *t-26* can be fired simultaneously. By firing of these transitions the attributes *X* of the places *p-18, p-29, p-30* and *p-35* refer to *B-A, B:x;B:y=2, A(x-x1), (x-x1)(x-x2),* respectively. Now, the transitions *t-12, t-14, t-21* and *t-31* can be fired simultaneously.

By firing of the transitions *t-12, t-14, t-21* and *t-31* the attributes *X* of the places *p-20 (D), p-21, p-33* and *p-39* refer to *(B-A)/(x3-x1), C-B, y1+A(x-x1), (x-x1)(x-x2),* respectively. Now, the transitions *t-15* and *t-28* can be fired simultaneously.

By firing of the transitions *t-15* and *t-28* the attributes *X* of the places *p-23 (E)* and *p-37* refer to *(C-B)/(x2-x4),* respectively. Now, the transitions *t-16* and *t-27* can be fired simultaneously.

By firing of the transitions *t-16* and *t-27* the attributes *X* of the places *p-25* and *p-41* refer to *(D-E), y1+A(x-x1)+D(x-x1)(x-x2),* respectively. Note that *(D-E)* corrects previous mid result *(C-B)/(x2-x4).* At this moment, the transitions *t-16* and *t-27* can be fired parallel.

After previous sequences of the parallel firing of the transitions the transition *t-17* is enabled, only. By firing of this transition the place *p-26 (R)* refers to *(D-E)/(x1-x4)* and the transition *t-32 is* enabled, only.

By firing of the transition *t-32* the place p-40 refers to *R(x-x1)(x-x2)(x-x3)* and the transition t-33 is enabled, only. By firing of this transition the attribute *X* of the place *p-42* has a value equal to *N(x)* (calculated as (4),(5),(6)). After firing of the transition *t-33* the net reached dead-node and there is no more enabled transitions in the net.

The final state (dead-node), after all previously described sequences of the transition firing is shown in Figure 3. *UPN model of Newton's interpolation algorithm, dead-node reached, left part.* and Figure 4. *UPN model of Newton's interpolation algorithm, dead-node reached, right part.*Only the place *p-42* has $\mu_{(p\text{-}42)}=1$ and all other places have marking equal to zero. The all calculated coefficients, mid results and final result are presented as appropriate *X* value of the places as described in firing sequences.

There are several transition functions we have used in the UPN model: *SUB* (subtraction, tfl:2), *DIV* (tfl:2, division), *COPY* (tfl:1, copy), *ADD* (tfl:2, addition) *MUL* (tfl:2,

multiplication), *none* (tfl:0, no transition function). We also have used front functions with standard arcs (such as $F^S(t\text{-}6, p16)=2$) and back functions (such as $B(t\text{-}26, p\text{-}35)=2$).

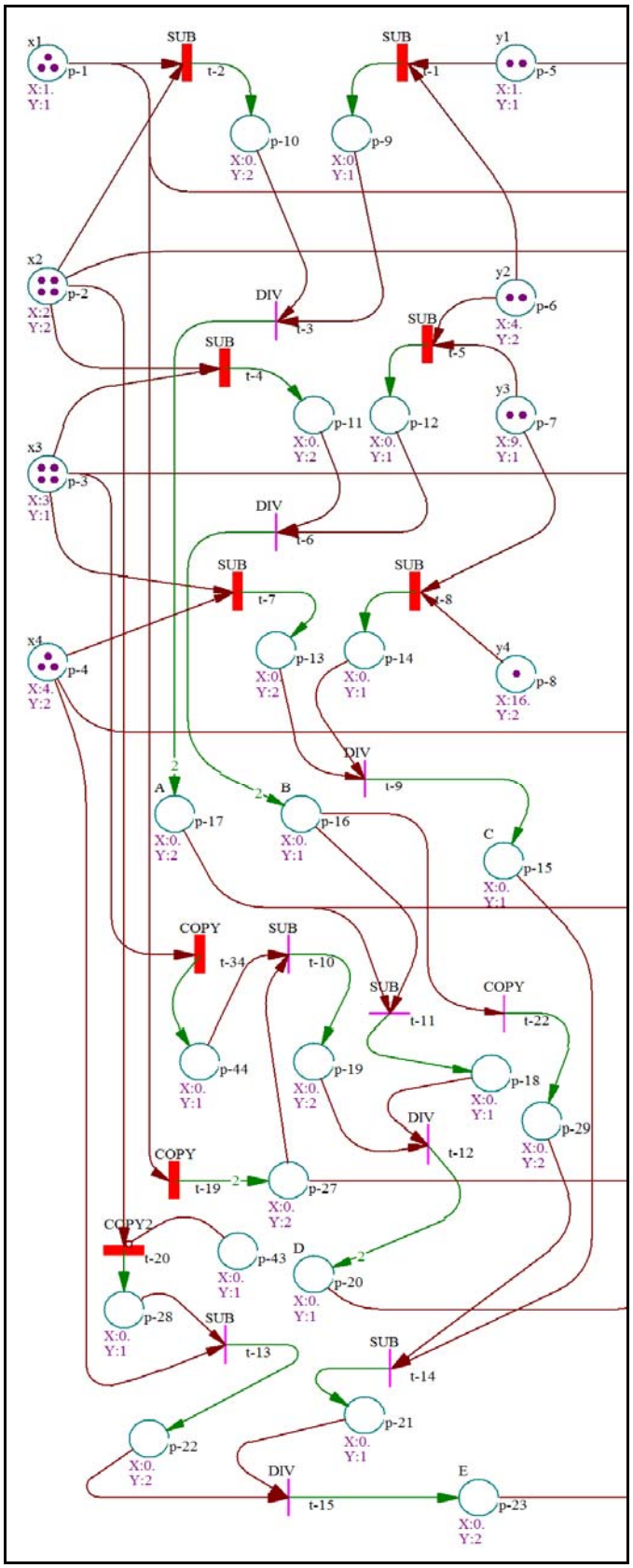


Figure 1. UPN model of Newton's interpolation algorithm, initial marking, left part.

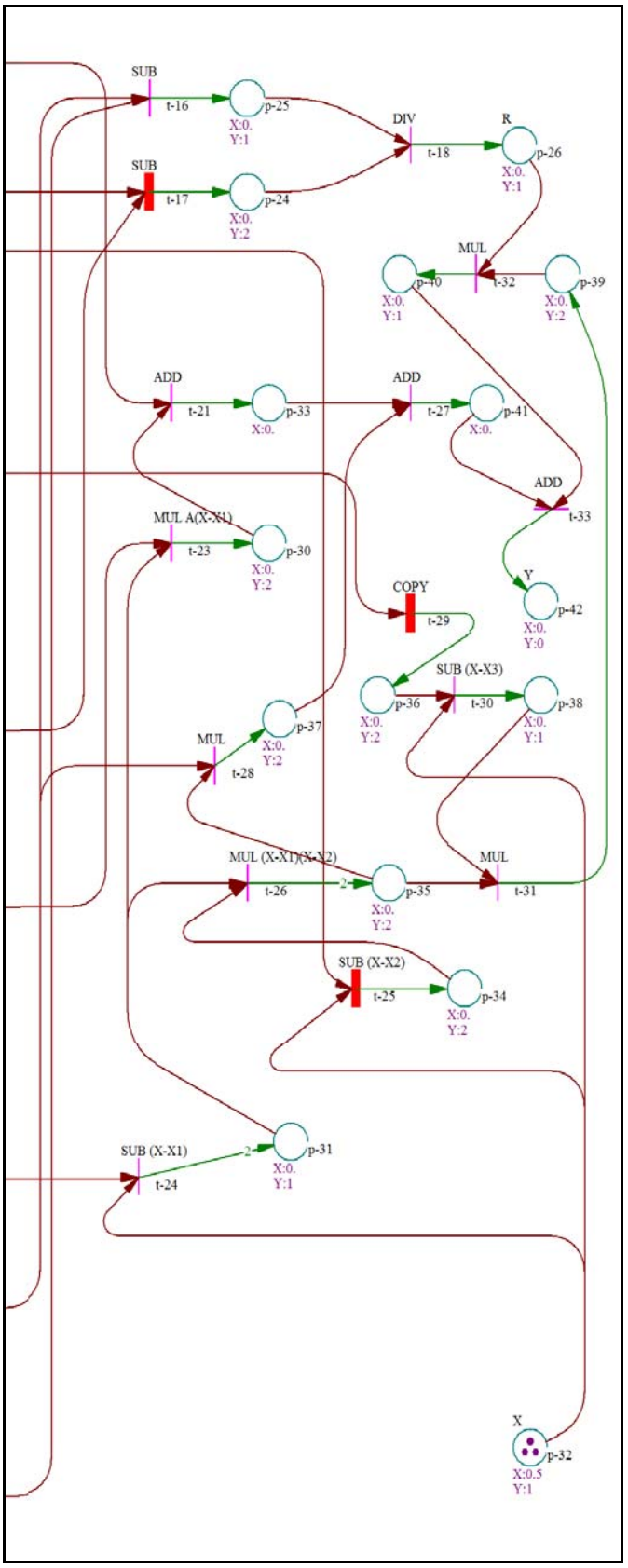


Figure 2. UPN model of Newton's interpolation algorithm, initial marking, right part.

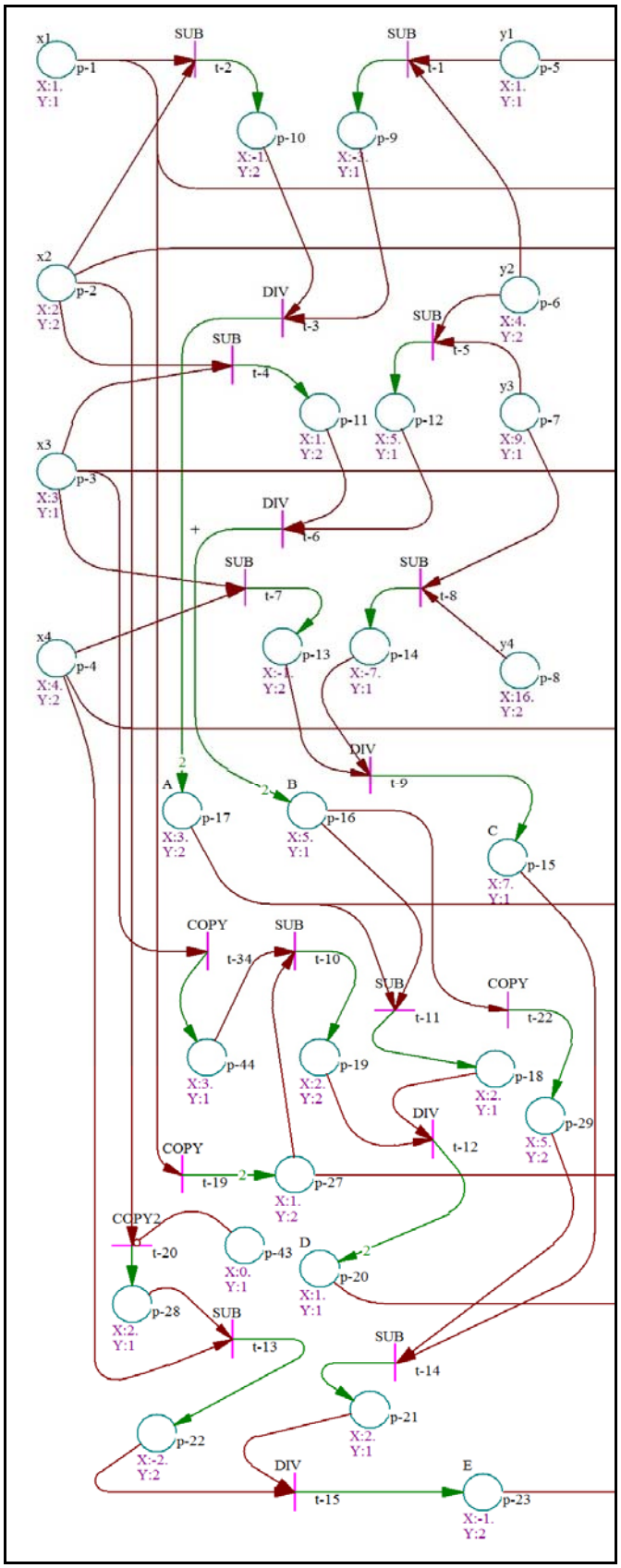


Figure 3. UPN model of Newton's interpolation algorithm, dead-node reached, left part.

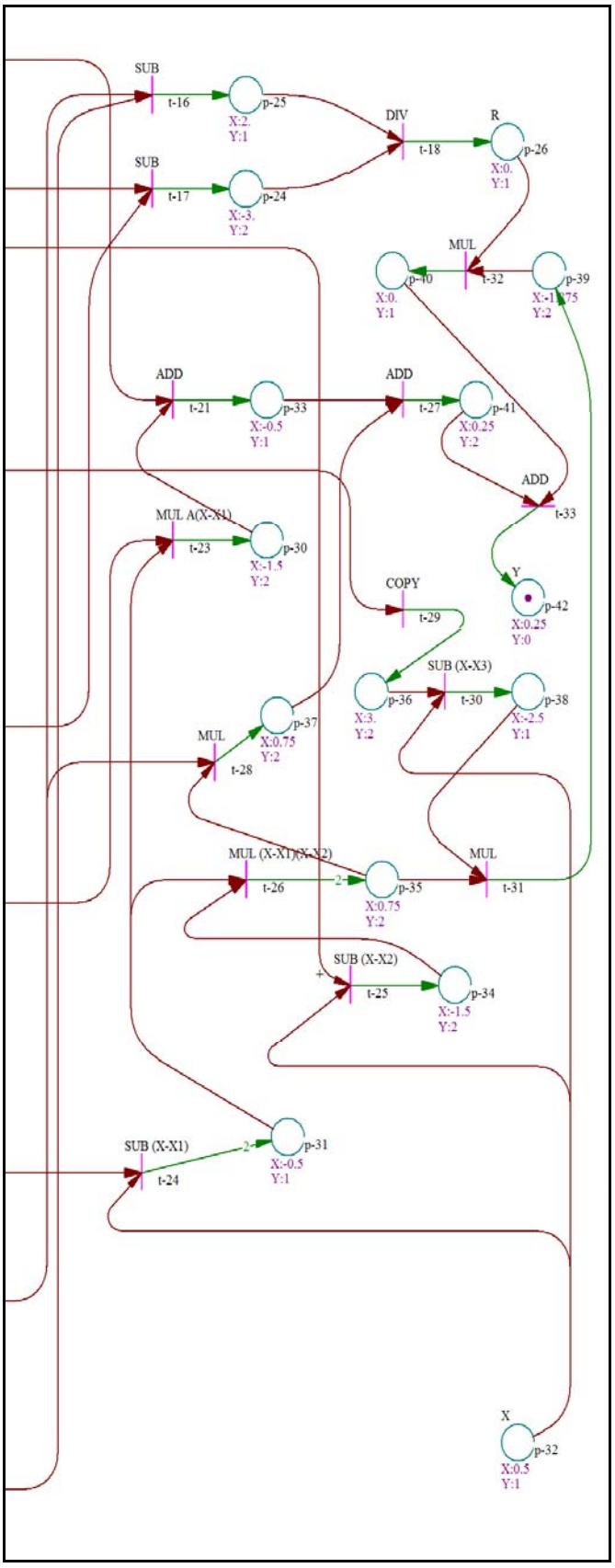
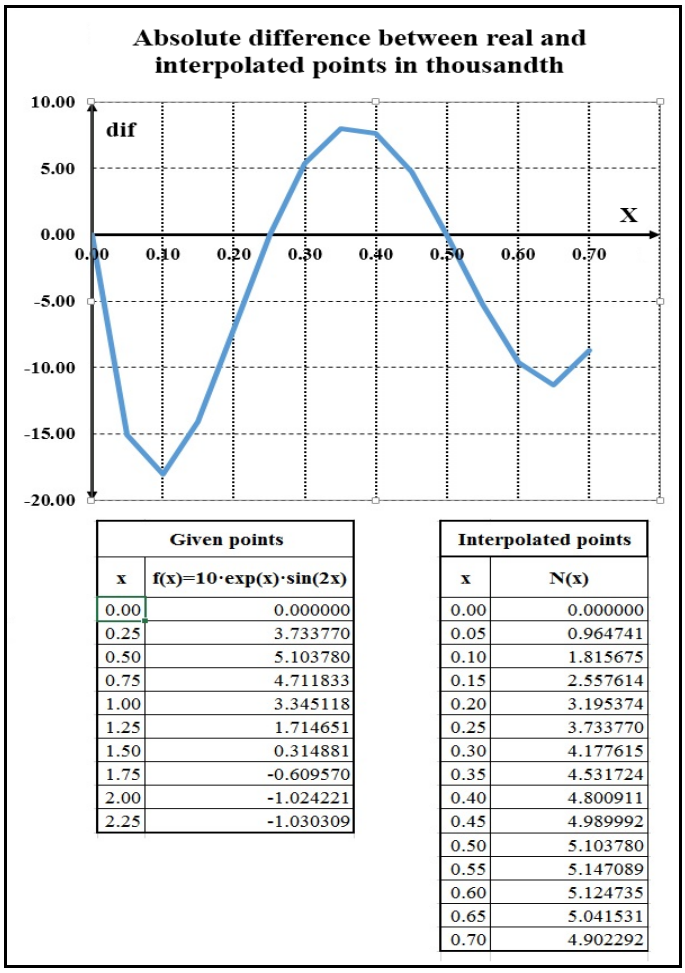


Figure 4. UPN model of Newton's interpolation algorithm, dead-node reached, right part.



Given points	
x	f(x)=10·exp(x)·sin(2x)
0.00	0.000000
0.25	3.733770
0.50	5.103780
0.75	4.711833
1.00	3.345118
1.25	1.714651
1.50	0.314881
1.75	-0.609570
2.00	-1.024221
2.25	-1.030309

Interpolated points	
x	N(x)
0.00	0.000000
0.05	0.964741
0.10	1.815675
0.15	2.557614
0.20	3.195374
0.25	3.733770
0.30	4.177615
0.35	4.531724
0.40	4.800911
0.45	4.989992
0.50	5.103780
0.55	5.147089
0.60	5.124735
0.65	5.041531
0.70	4.902292

Figure 5. Simulation results.

IV. THE UPN MODEL ANALYSIS

The UPN models analysis was done through two phases. The first phase refers to transforming observed UPN model shown in this paper in a software program using *C++* language. The second one refers to checking behavior of the UPN model shown in previous chapter by executing the software program which refers to the models presented in this paper. This software program has been tested for various values of input parameters. The data of 10 given points was placed in an input file. According to these points we calculate appropriate Newton's interpolation polynomial. For some x input value which belong to interval determined by x coordinates of the given points, we calculate interpolated y value, i.e. value of appropriated Newton's interpolation polynomial. Some of obtained simulation results are shown on Figure 5. *Simulation results.* These results refer to the input parameters as follows: number of given input points is 10, and we calculate interpolated value for our 15 points. The validity of the UPN models and its parallelism and synchronization at register transfer level presented in this paper are confirmed by the results of the implemented software program.

V. CONCLUSION

This paper presents Upgraded Petri net models of Newton's interpolation algorithm with its parallelism and synchronization at register transfer level. For given input points the UPN through its execution calculated Newton's interpolation algorithm for x value which belongs to interval determined by x values of the given points. The UPN model in this paper represents parallelism and synchronization in calculation of the Newton's interpolation polynomial. A parallelism and synchronization are presented in the models by using UPN formal theory were shown in this paper. The UPN model was developed from the initial version to the final version through interactive monitoring of its execution from the initial marking up to the dead node. After careful analysis, the UPN model is transformed into the real software program. Suitability of given UPN model was checked by execution of the UPN model for given input data. Original software for modeling and simulations of UPN, PeM (Petri Net Manager), is developed and used for all models described in this paper.

ACKNOWLEDGEMENTS

This research was supported by the Serbian Ministry of Education and Science, Republic of Serbia, Project no. III 44006

REFERENCES

[1] Michel Diaz, "Petri Nets: Fundamental Models, Verification and Applications", John Wiley & Sons 2010.

[2] Girault, C. and Valk, R., "Petri Nets for Systems Engineering - A Guide to Modeling, Verification, and Applications", Springer-Verlag, Berlin, Heidelberg, New York, 2003.

[3] Perica Štrbac, Gradimir V. Milovanović, "Upgraded Petri net model and analysis of adaptive and static arithmetic coding", Elsevier: Mathematical and Computer Modelling Vol. 58, http://dx.doi.org/10.1016/j.mcm.2013.06.001, pp. 1548–1562, 2013.

[4] Xiaofeng Wang, Jovana Dzunic, Tie Zhang, "On an efficient family of derivative free three-point methods for solving nonlinear equations", Elsevier: Applied Mathematics and Computation, Vol. 219, pp. 1749–1760, 2012.

[5] Mohammad Masjed-Jamei, Gradimir V. Milovanović, M.A. Jafari, "Explicit forms of weighted quadrature rules with geometric nodes," Elsevier: Mathematical and Computer Modelling, Vol. 53, doi:10.1016/j.mcm.2010.11.076, pp. 1133–1139. 2011.

[6] Joachim von zur Gathen, Jürgen Gerhard, "Fast polynomial evaluation and interpolation", Modern Computer Algebra, chapter, Vol. 1, http://dx.doi.org/10.1017/CBO9781139856065.013, pp. 295-312, 2013.

[7] Rohitha Goonatilake, Vicente O. Ruiz "On Polynomial interpolation approximations: Numerical vs. statistical techniques", Journal of Modern Methods in Numerical Mathematics, Vol. 5, no. 1, pp. 17–27, 2014.

[8] Samir M. Koriem, T.E. Dabbous, W.S. EI-Kilani. A New Petri Net Modeling Technique for the Performance Analysis of Discrete Event Dynamic Systems", Elsevier: The Journal of Systems and Software, Vol. 72, Issue 3, pp. 335–348, 2004.

[9] Shuang'e Zhouab & Guoping Xiong, "Task Synchronization Process based on Petri Net ", Intelligent Automation & Soft Computing, DOI:10.1080/10798587.2011.10643181, Volume 17, Issue 6, pp. 713-724, 2011.